A User Study Contrasting 2D Unsteady Vector Field Visualization Techniques

A Thesis

Presented in Partial Fulfillment of the Requirements for Graduation with Distinction in the Department of Computer Science and Engineering at The Ohio State University

By
Nathan Andrysco

.....

The Ohio State University
2005

ABSTRACT

Scientific visualization is very popular today among many different fields. Its purpose is to take data and represent it visually so that a person can better understand what is occurring with the data. This idea can be as simple as a pie chart or as complex as drawing the beating human heart with data taken from a MRI scan.

One particular area of scientific visualization is the representation of flow data. Many methods have been researched and published on how to visualize different types of flow fields, but only a few papers have been published that have attempted to compare and contrast the effectiveness of the various methods.

For the study, five different flow field visualization methods are compared – LIC, pathline, streamline, streakline, and hedgehog. A framework was created to not only display each of the methods but also allow the user to interact with them. The user is shown an unsteady flow field using one of the five visualization methods. He may view the flow field at any time step using a slider or hitting a button to animate the flow. The user is asked to make a response on two different types of tests. The first one asks the user to advect a particle to a circle. The angular error is recorded into a file. The second test asks the user to determine where a massless particle might have begun. The number of clicks it takes the user to correctly identify where the particle began is recorded.

Statistical analysis was performed on the data to determine which visualization method is the "best". Unfortunately after using 95% confidence intervals, no statistical conclusions could be drawn about what were the better

methods. However, it could be stated that the test method could be refined to produce better results. Or if the test method is fine, it just might be the fact that even with the best visualization methods out there that a person is unable to fully comprehend an unsteady flow field.

ACKNOWLEDGEMENTS

I would first like to thank my advisor for this project, Han-Wei Shen. Not only has he provided me with a great deal of help, but his class is what first made me interested in computer graphics (and ultimately to write this thesis).

Roger Crawfis for also advising me on the project and helping obtain users for my framework.

Liya Li for providing “advection” code, which greatly speed up the programming.

And to everyone that used my program and reported their results and comments.

VITA

1983 ……..	Born
2001-2005 …….	Undergraduate Student Computer Science and Engineering The Ohio State University
2005-Present …….	Graduate Student, Ph.D. Program Computer Science Purdue University

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The goal of this project is to classify different unsteady vector field visualization methods. Hedgehog, streamline, pathline, streakline, and LIC were the methods chosen for this study.

- Hedgehogs represent the data by drawing oriented scaled lines along the direction of the local vector [8].
- Streamlines are defined as curves tangent to the velocity field in every point [8]. In other words, they represent the path a massless particle would travel over one time step.
- Pathlines are obtained in a real world environment by putting small objects into the flow field and exposing a photograph for a longer time. The traces of the objects in the photograph represent the path of the particle over time [1]. Pathlines are represented on a computer by drawing curves that trace the path of a particle over time.
- In a real world environment, streaklines are produced by continuously injecting a material like smoke or little hydrogen bubbles into a flow field at certain seed points. The streaklines are the traces of these particles [1]. To represent streaklines on a computer, points are drawn to represent the bubbles or smoke.
- LIC, or Line Integral Convolution, is a texture based visualization method for flow fields. Textures are produced by taking a white noise background and

then “smearing” it along the direction of the vector fields. The resulting image represents the vector field at that time step [2].

Using these techniques a person is given a visual representation of their vector field data. What these techniques hope to do is show some underlying phenomena in the data. This phenomenon is also called critical points. Critical points can be formally defined as points where the magnitude of a vector vanishes. There are many types of critical points. They are characterized by the behavior of the nearby tangent curves.

- A repelling node is categorized by having all nearby vectors point away from the critical point. Another way to think about this critical point is that the repelling node pushes away all vector fields in its nearby vicinity. This type of critical point is also called a source.
- An attracting node, or a sink, is the opposite of a repelling node. All nearby vectors are pulled towards the critical point.
- The vector fields around a center critical point will be represented visually as a bunch of congruent circles with centers at the critical point.
- A repelling focus is the combination of a source critical point and a center critical point. The picture that results from the vector fields will look like a swirl spiraling outward from the critical point.
- An attracting focus is the combination of a sink and a center. This will look like a swirl spiraling inward from the critical point.
- A saddle point combines both a sink and source. On one axis the critical point repels the vector fields. In the other direction the saddle point repels the

fields [3].

A visualization method is considered good if the vector field is visually represented so that a person can understand what is happening in the data. A person should be able to identify where critical points exist and what type critical point it is. He should also be able to accurately advect a particle forward in time [6]. The focus of this thesis is to see which method helps a user most accurately advect a particle. The following sections will describe the test framework and the tests themselves.

CHAPTER 2

PREVIOUS WORK

Only three papers have been previously published on this subject. The first one is an initial study by David Laidlaw of Brown University. The study attempted to compare six visualization methods on steady flow fields. The visualization methods were GRID, JIT, LIT, LIC, OSTR, and GSTR. A framework was created and users were asked to interact with the program. The users were tested on both how quickly and how accurately they responded. Three tests were conducted. The first test asked the user to identify a critical point. The next test placed a dot on an otherwise blank screen. The user was then shown the steady flow field and asked to identify what kind of critical point was located where the dot previously was. The last test asked the user to advect a point inside of a circle to the boundary of the circle. After his tests were completed, he was unable to make any statistical conclusions using 95% confidence intervals [6].

A second paper by David Laidlaw extended his previous paper. The same framework and statistics were used. The big difference was in his testing methodology. He created more test cases and tested more users. By doing so he was able to make statistical conclusions [7].

The last paper is an article entitled "User Studies: Why, How, When?" It explains why user studies of visualization methods are useful and how they should be conducted [5].

CHAPTER 3

FRAMEWORK

The framework is programmed in the Windows environment. The graphics programming is done in OpenGL. The GUI library that was chosen is FLTK. FLTK is used because it is easy to setup an OpenGL window and to add other widgets with callbacks (such as a slider). FLTK is very similar to a simpler toolkit called GLUT. GLUT was not used because the programmer is unable to add widgets or create multiple windows. Windows programming was considered, but FLTK is much simpler for a novice GUI programmer.

The graphical loop of the program is where most of the other functions are called. The only functions not called from the main loop are all the callbacks (mouse, keyboard, widget). The first function that is called from this display loop is the function to display the selected visualization method. The visualization method is selected randomly from the five types explained earlier in the paper.

Hedgehog is the simplest method. It merely draws a structured grid of points. At each point a line is attached. The line is oriented in the direction of the vector at that seed point. The length of the line is proportional to the magnitude of the vector. Figure 1 is an example of what hedgehogs will look like.

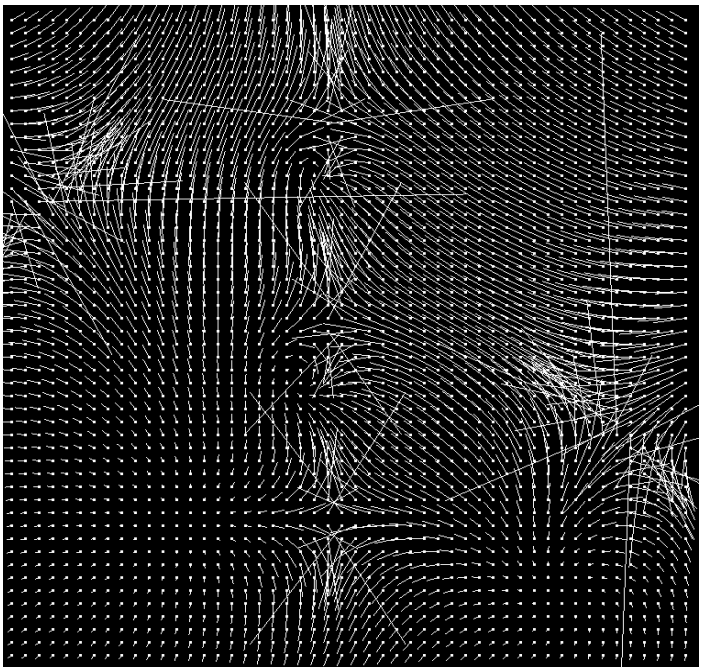

Figure 1: Hedgehog Screenshot

Pathlines and streamlines look very similar. Both methods take numerous seed points and trace paths from them. The position of the seed points are randomly selected. The number of seed points used is defined by the programmer. Both display functions call an advection function many times to calculate points along their curves. The advection code is written by Liya Li, a Ph. D. student at The Ohio State University. The difference between pathlines and streamlines is in how the points of the curve are calculated. Streamlines will only look at vector data at one point in time. For instance, a particle is placed in the vector field at a specific time step. This particle will be advected in this vector field until it is told to stop. Pathlines, however, are a trace of a particle over multiple time steps. This means that a particle starts at a time step, t, and is tracked over the vector fields at time = t for one time period. When the next time period begins, the particle will start where it left off at pervious time period. But instead of tracking the particle at the vector field

associated with time = t, the particle will be tracked in the vector field at time = t + 1. This continues until the number of time steps (specified by the programmer) has been completed. Once all the points along the curves are calculated, the curves can be drawn onto the computer screen. Figure 2 shows an example of what pathlines look like, whereas Figure 3 shows an example of what streamlines look like.

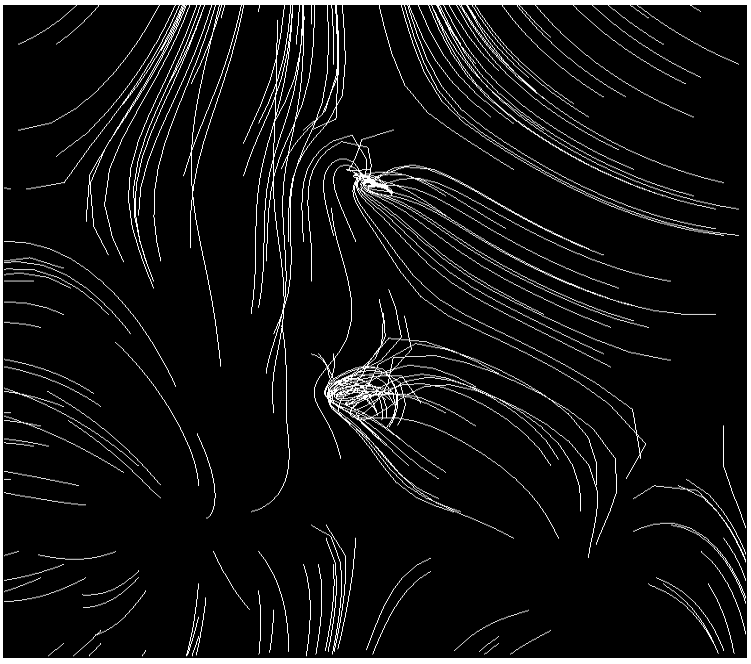

Figure 2: Pathline Screenshot

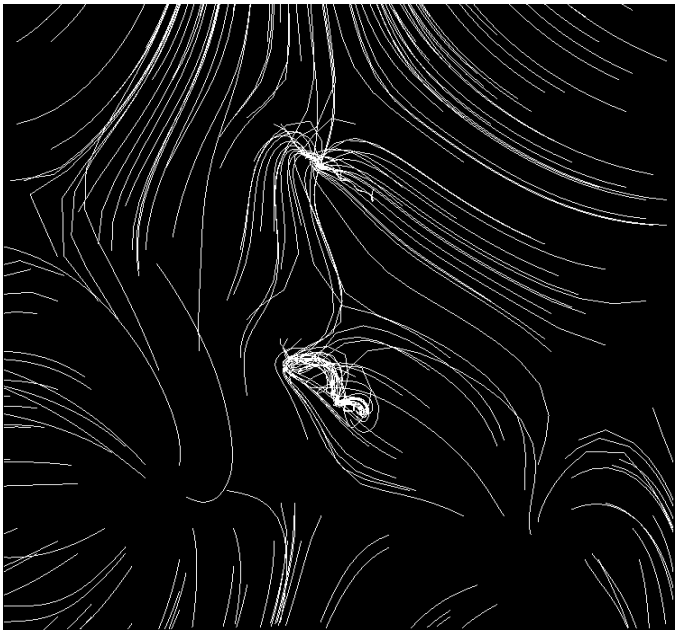

Figure 3: Streamline Screenshot

Streaklines are calculated in a way similar to pathlines. The only difference is that particles are continuously injected into the flow field. For instance, at time step = t, twenty particles are placed into the field at twenty seed points (a particle per seed point). The path of these particles is calculated until time = t + 1. At the next time period, the twenty particles that originated at time = t are further advected. However, twenty new particles are injected into the vector field at the seed points. These particles also need to be traced for a time step, this time starting at time = t + 1. So at time = t + 1, there are forty particles that need to be tracked. At time = t + 2, there will be sixty particles. This continues until the max time step is reached (specified by the programmer). Once all the paths of these particles are calculated, points are drawn to display the paths of the curve. Figure 4 is an example of what streaklines look like.

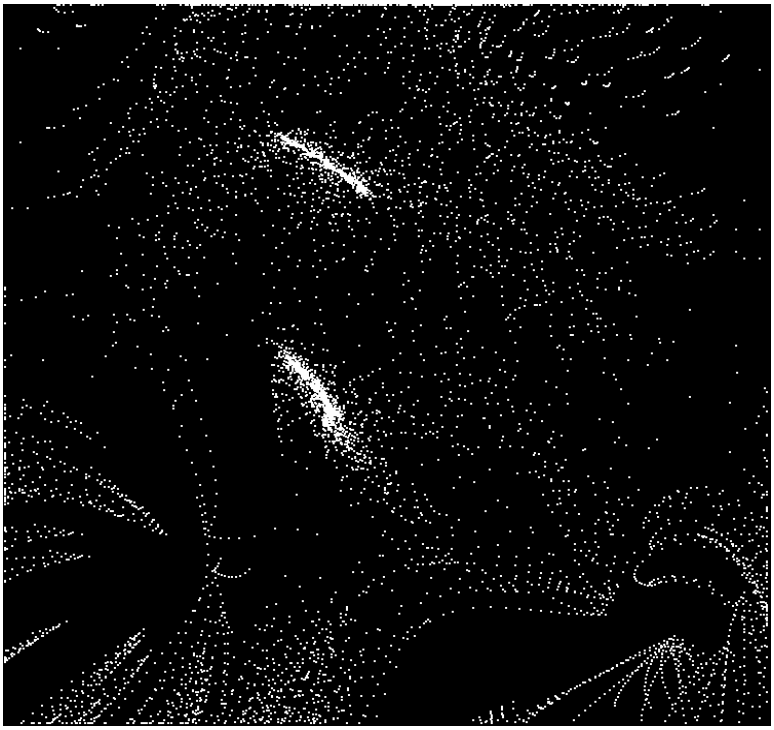

Figure 4: Streakline Screenshot

The final visualization method, LIC, is texture based. The LIC drawing function calls a class created by Han-Wei Shen, a professor at The Ohio State University. The LIC class generates a white noise image. This image is then smeared in the direction of the vector fields at a given time step. The resulting image is copied into texture memory and decaled onto an OpenGL quad. Figure 5 is an example of what a LIC image looks like.

Figure 5: LIC Screenshot

After the visualization method has been drawn to the screen, the user test is selected. All tests are defined by the programmer. The class created for this purpose easily allows another programmer to add his own tests that he wishes to use in his own studies. What is drawn to screen is defined by the programmer in the test class. The program waits for the user to respond and validates that it is a valid response (again, as with all functions called by the test, it is defined by the programmer). If a valid response is made, the program checks to see if the test is finished. If not, the program waits for another response. If the test is finished, the result is output to a file and a new test and visualization method are randomly selected. Once a new test is selected, a window will pop-up and describe to the user what he is supposed to do. Also, the type of visualization will be described to him. Further descriptions of the tests involved in this study will follow in the next section.

This process continues until all tests defined by the programmer are completed. Once all tests are completed, the program exits.

CHAPTER 4

TEST METHOD

Two types of tests were used in this user study. The first test asked the user to advect a point to a boundary. In the second test, the user had to determine where an already advected point began. The user is given a few controls to help him better in visualizing the unsteady flow field. A slider is provided to the right of the OpenGL display window that allows the user to change what time step he is currently viewing. The user may also hit the 'a' key to animate the unsteady flow field (or hit the 'a' key again to stop animation). The amount of time each slice will be displayed before the next one is shown can also be controlled with a slider that is to the right of the OpenGL display window. Giving the user control of what time slice is being displayed should help him in making a response when interacting with the tests.

The first test draws a circle and a point inside the circle. Figure 6 shows an example of what the user may see. The user is asked to visualize the path of the point if the point was to be advected. The next step is for the user to click on the outline of the circle where he believes the path first intersects the circle. If the user fails to click close to the outline of the circle, no response is recorded. The user can not go to the next test until he has clicked on the outline. Once the user has clicked on the outline of the circle of where he believes the particle will be advected to, the angular error is computed and output to a file. The angular error is the number of radians the user's click was away from where the actual particle was advected to. This test will be referred to as test one for the rest of the paper.

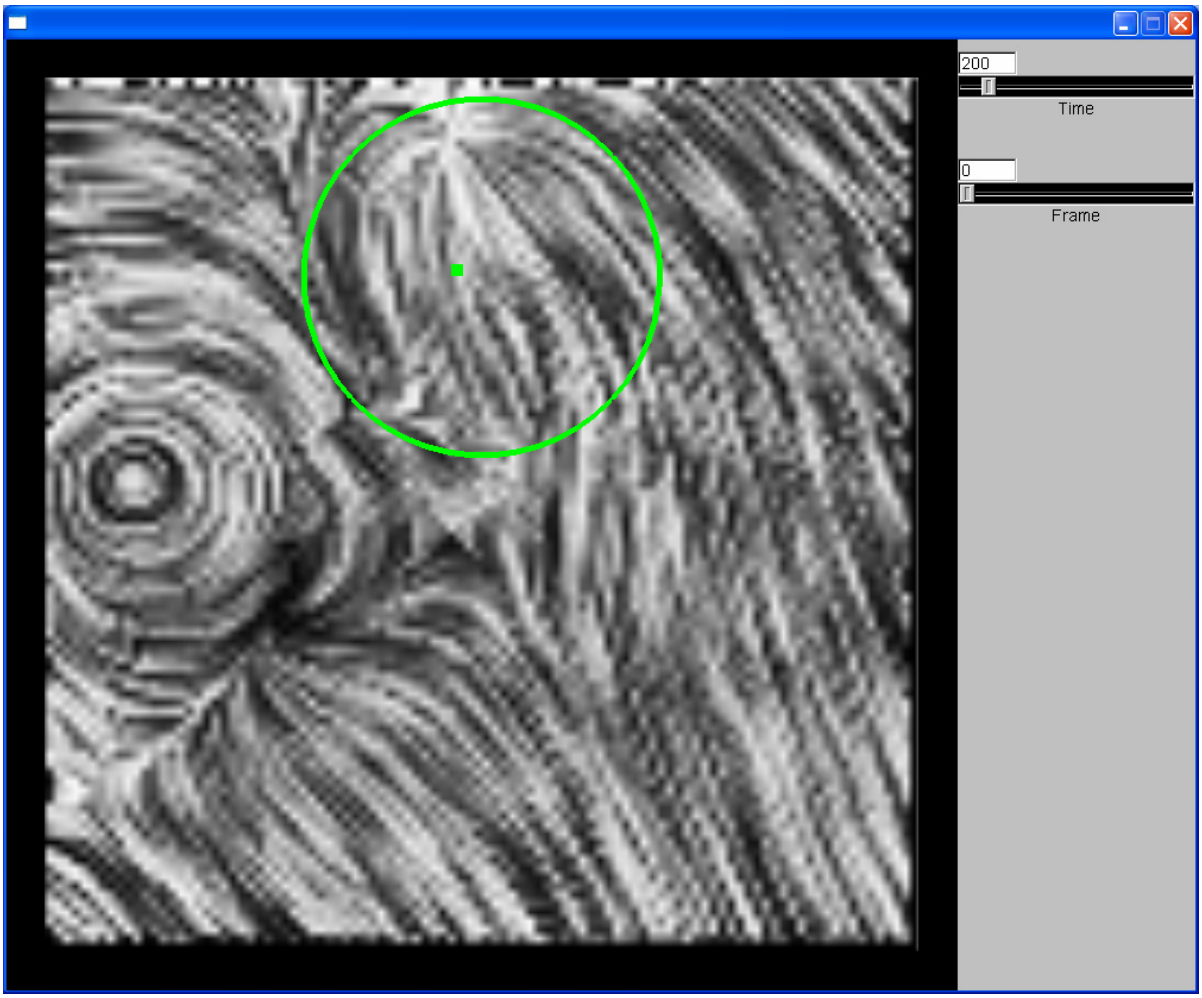

Figure 6: Test One Screenshot

The second test draws two concentric circles, with one having half the radius of the other. A point is drawn on the outline of the outer circle. Figure 7 is an example of what the user may see if he was doing this test. The user is asked to pick a point inside the inner circle that he believes will be advected to the point on the outer circle. If the point the user picks is within the inner circle and the point's advected path hits the outer circle's outline close enough to the drawn point, the test is finished. If the path of the user selected point does not come close enough to the drawn point, that incorrect path is displayed and the user must choose a new point. An example of what the user might be shown after an incorrect response is Figure 8.

If the intersection of the advected path and the outer circle is close enough to the drawn point, the results of the test is recorded to a file. The number of clicks it takes a person to correctly find a seed point that is advected to a place near to the drawn point on the circle is what is being recorded. One note, a maximum number of clicks (twenty) was implemented. This is to prevent a user who is having trouble with a test to be clicking away for a long time and never finishing. This type of test will be called test two for the rest of the paper.

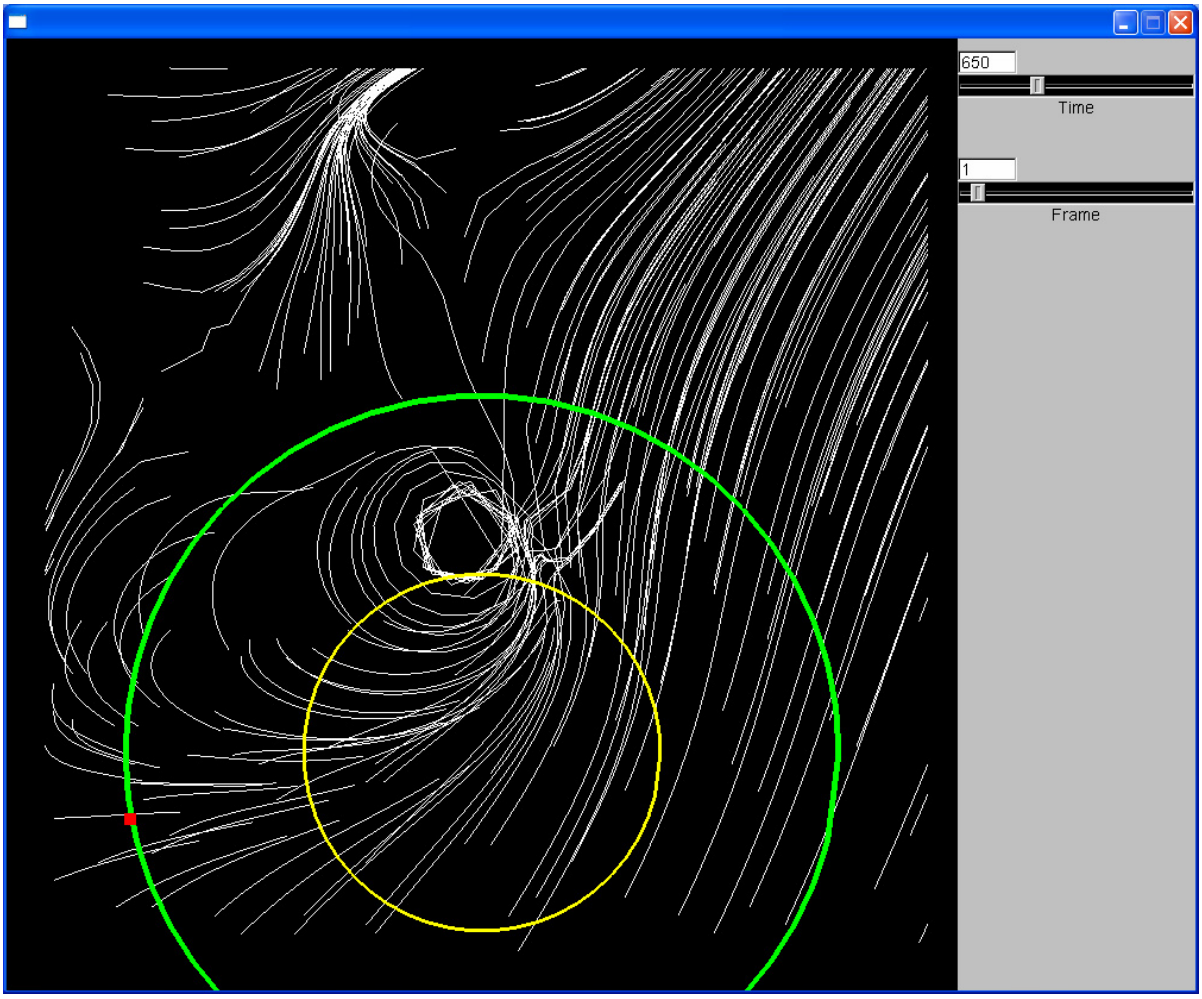

Figure 7: Test Two Screenshot – No Response

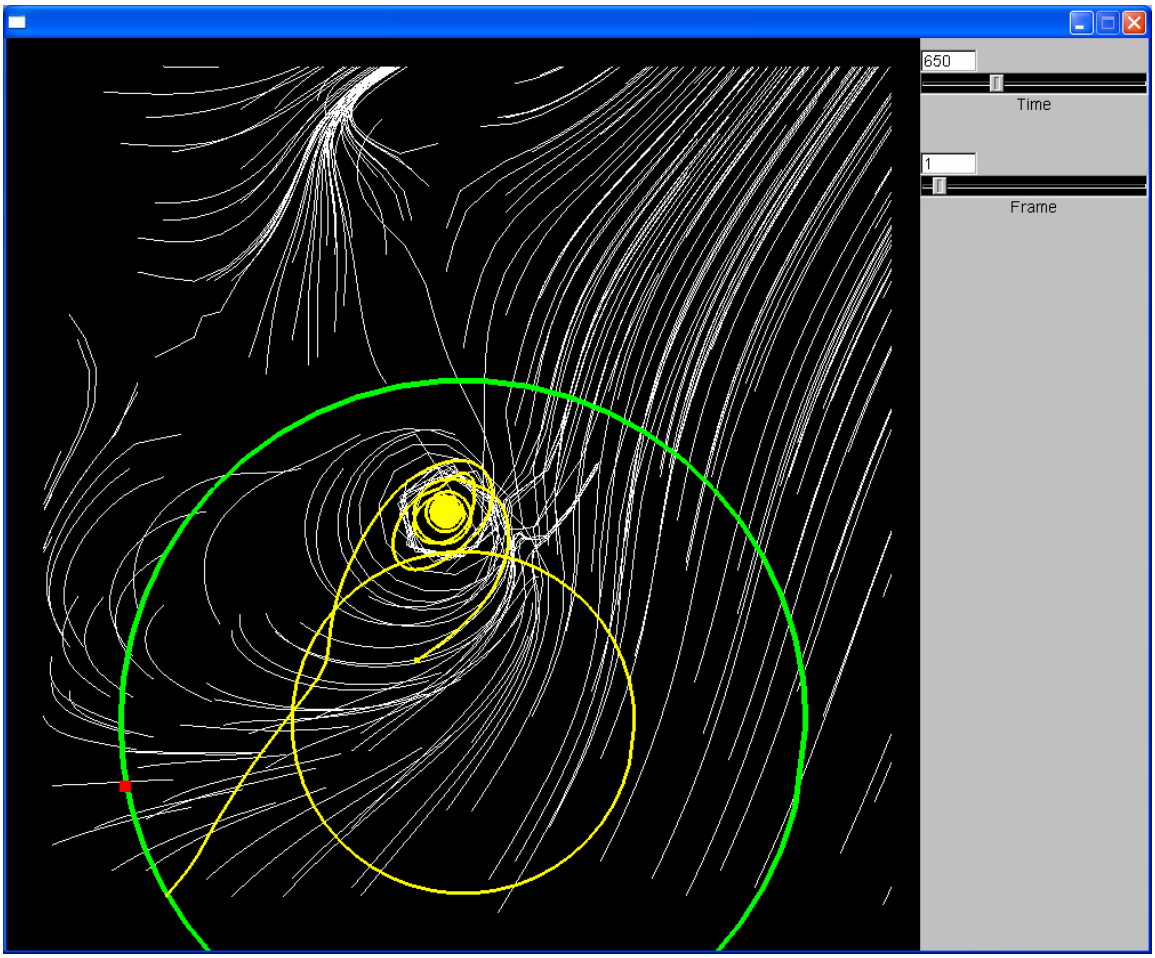

Figure 8: Test Two Screenshot – Incorrect Response

At the end of both tests, other information is written to the results file that helps with statistical analysis. The most important is the visualization method. This is obviously the most important because it allows the programmer to compare the different methods. Other variables such as size of the circles used, how many critical points were present in the data, and how close the circle is to nearby critical points are also written. It is the hope that these variables will give some kind of insight into how future tests can be constructed so that better results are achieved. For instance, say that after doing statistical analysis it is found that too large of a circle radius results in very poor responses. Using this information, future studies will be able to

obtain more reliable results.

Not only is the test type defined by the programmer, he also defines variables such as the size of the circle, where the circle will be placed, and so forth. The purpose having the programmer give the variables fixed values instead of having the variables randomly assigned was so that the programmer could setup the tests so they were not impossible or too easy. Problems occurred as a result of this methodology, which is explained in the Conclusions section.

CHAPTER 5

STATISTICAL ANALYSIS

95% confidence intervals were used to analyze the data. A visualization method or other variable is said to be statistically better if its interval does not overlap another interval and its interval is lower on the graph. More variance, or less reliable data, is said to occur when the interval is large.

For test two, the user was either able to quickly find a solution or had extreme difficulty. From talking with some of the users after they took the test, if they had difficulty with a specific test they would randomly click until a solution was found or the limit of maximum number of clicks was reached. This erroneous data is ignored by filtering responses that took more than fifteen clicks. Using that filter, twenty-eight out of 197 test results were discarded. So about 14% of the time the user had difficulty with the test.

All together, twenty-nine people took the test. This resulted in 433 tests completed, which includes the twenty-eight that were filtered.

CHAPTER 6

STATISTICAL RESULTS

The following graphs show the plotted confidence intervals of the data that was collected. Unfortunately, no statistical conclusions can be made. Conclusions that do not rely on statistics will follow in the next section of the paper.

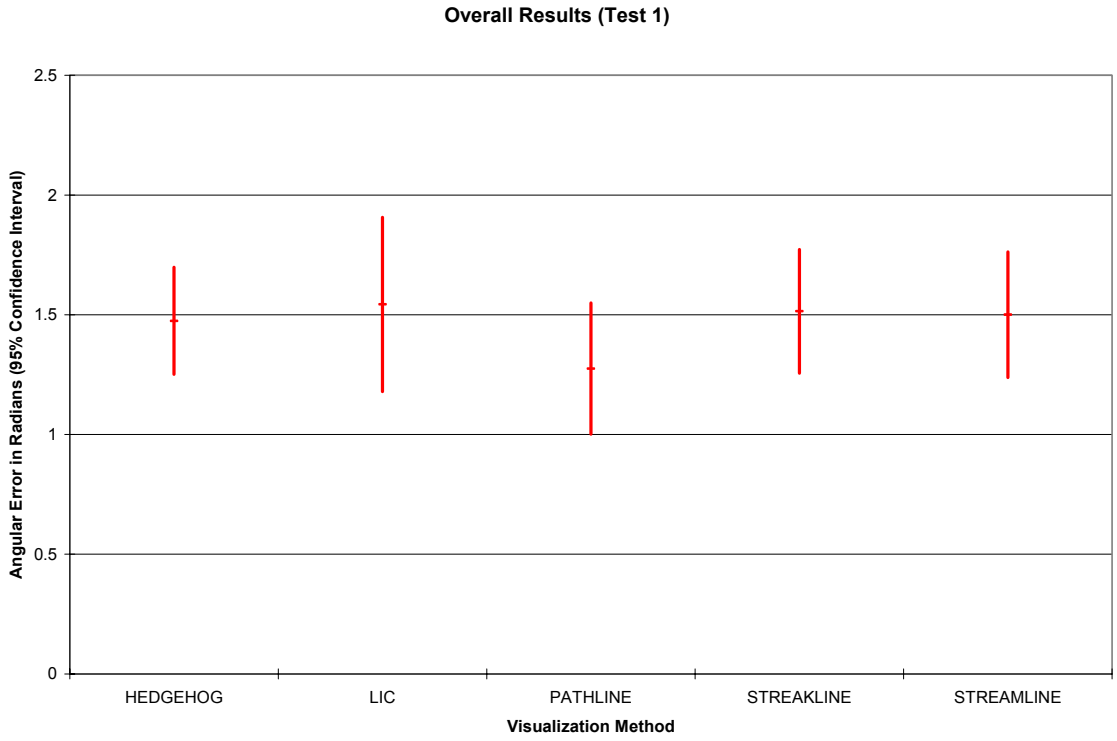

Figure 9: 95% confidence intervals of how each visualization method performed on test one.

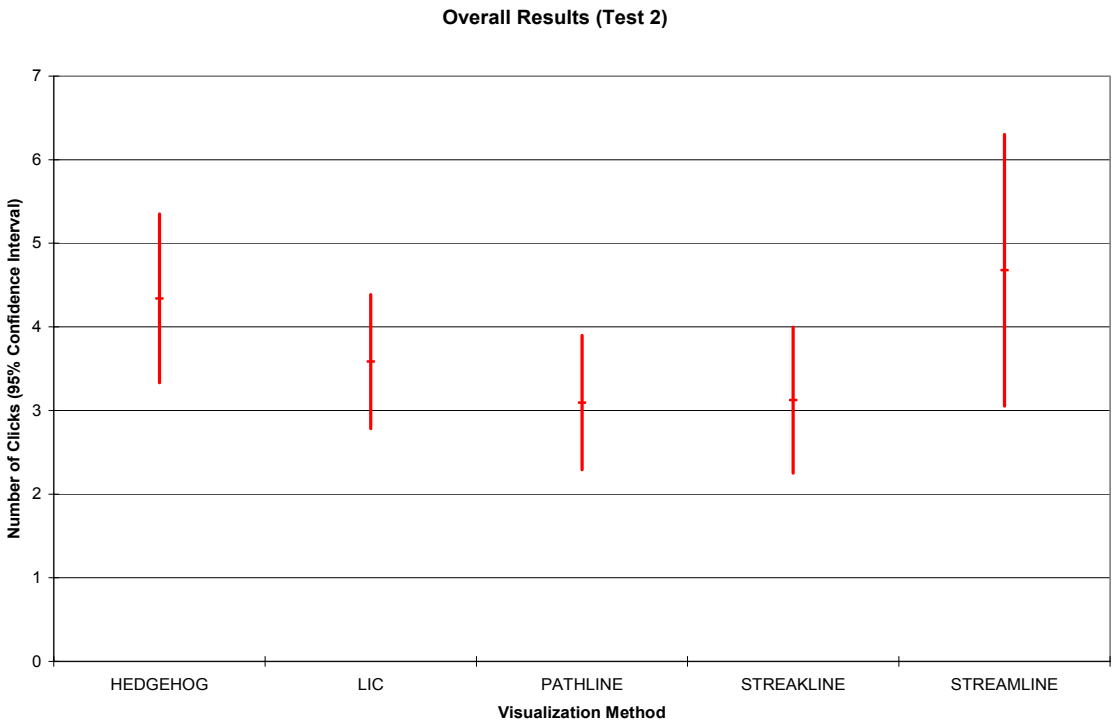

Figure 10: 95% confidence intervals of how each visualization method performed on test two.

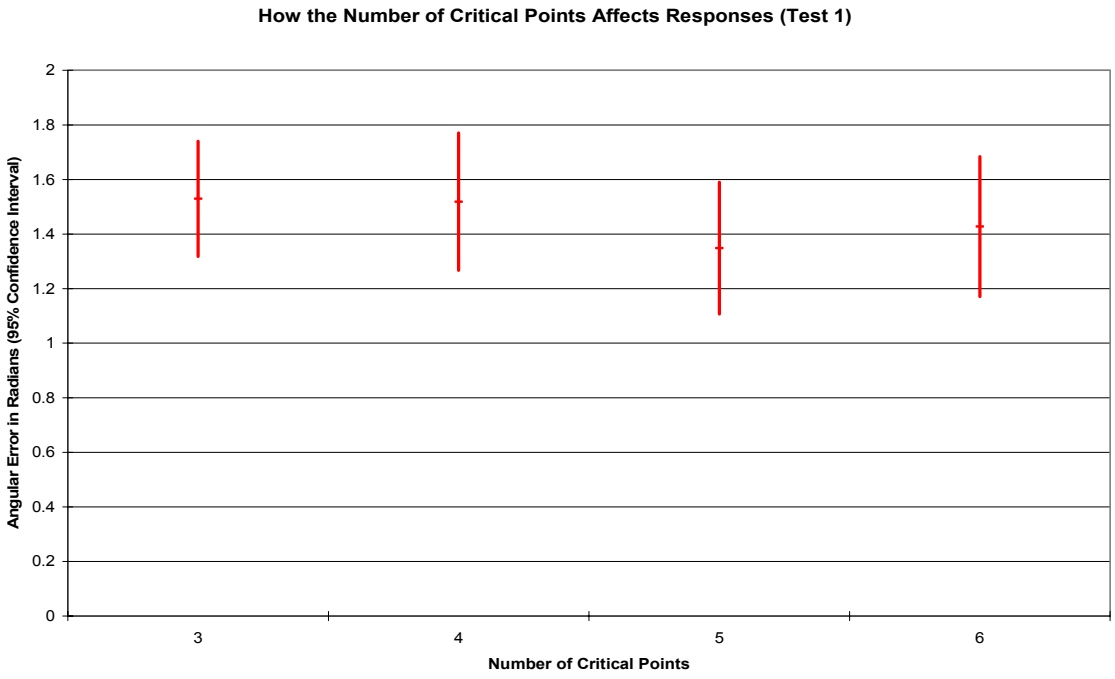

Figure 11: 95% confidence intervals of how the number of critical points in the data set affected the responses of users while using test one.

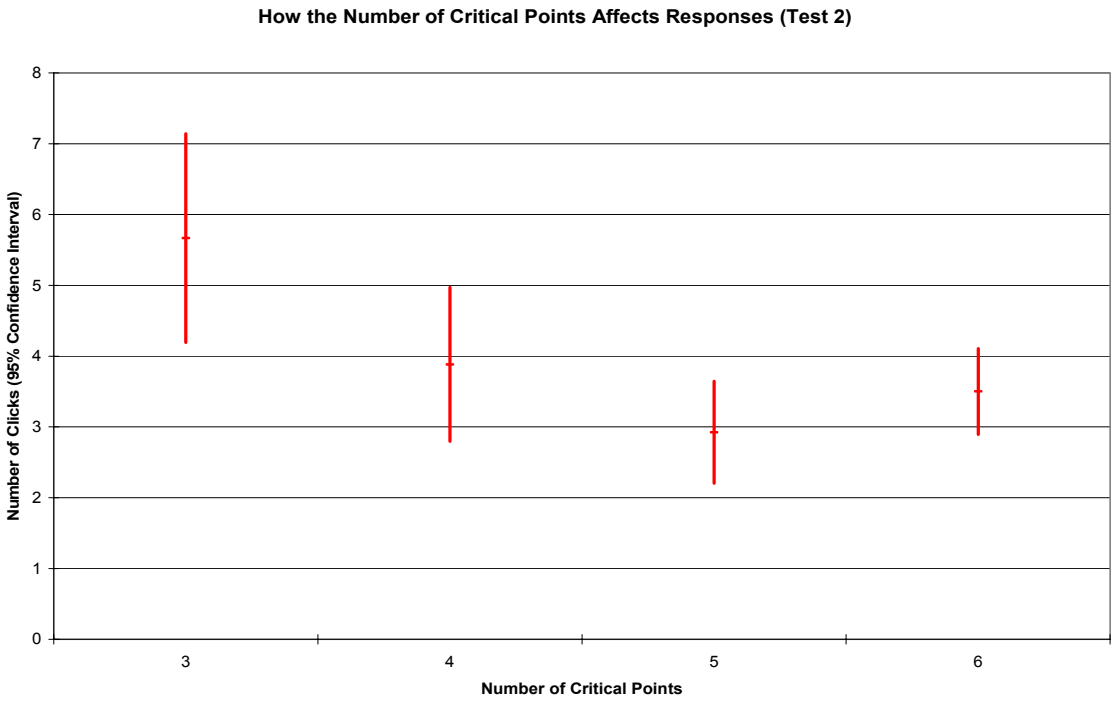

Figure 12: 95% confidence intervals of how the number of critical points in the data set affected the responses of users while using test two.

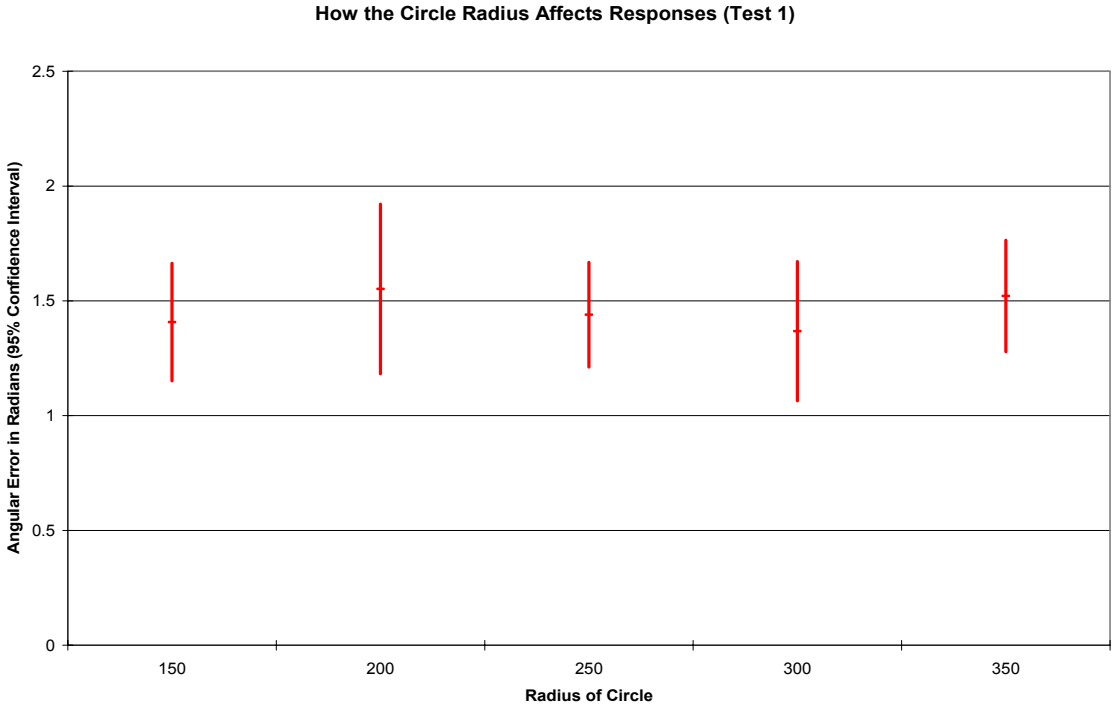

Figure 13: 95% confidence intervals of how large the radius of test one's boundary circle affected the responses of users.

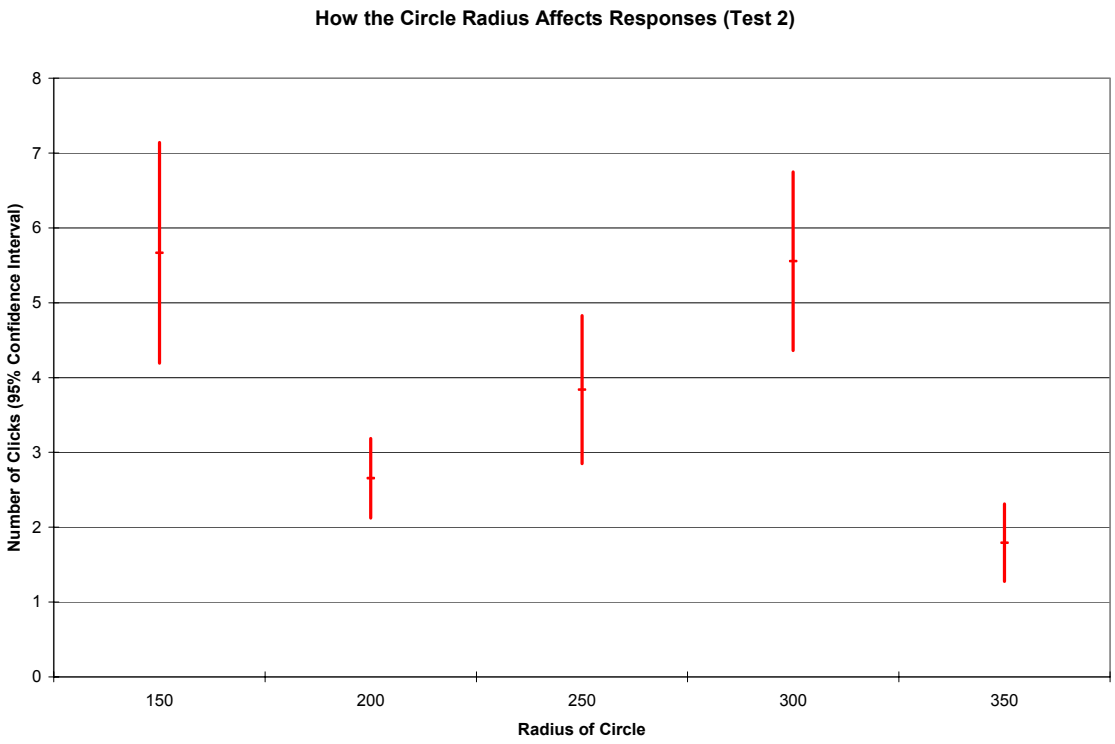

Figure 14: 95% confidence intervals of how large the radius of test two's outer circle affected the responses of users.

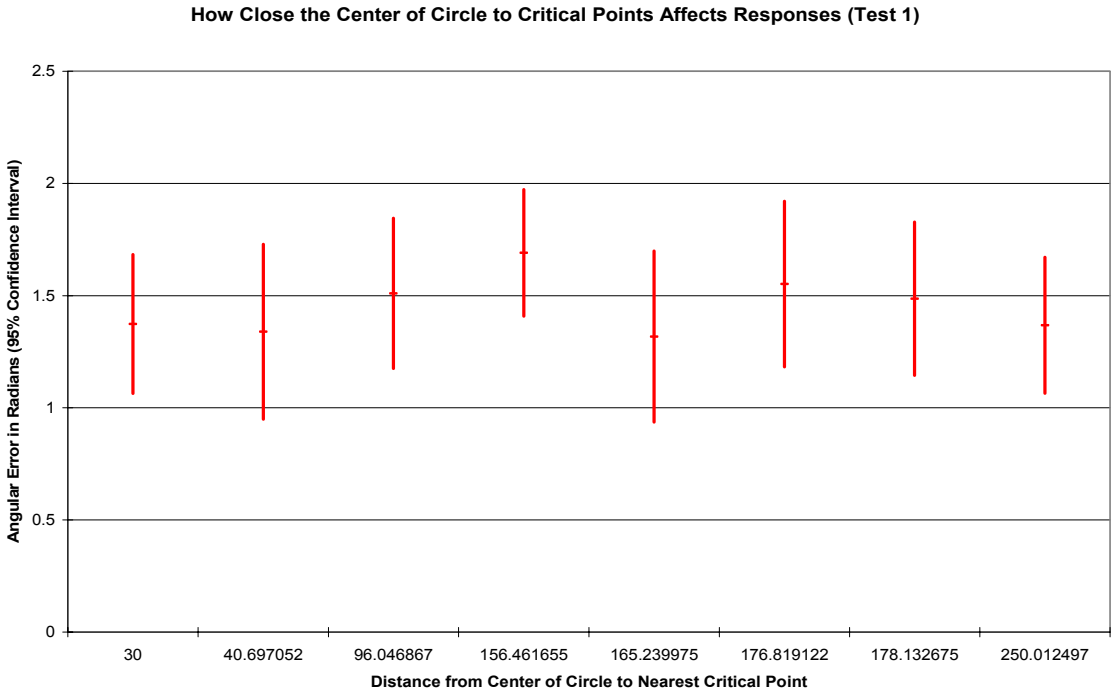

Figure 15: 95% confidence intervals of how close the center of the circle was to the nearest critical point and how this distance affected results for test one.

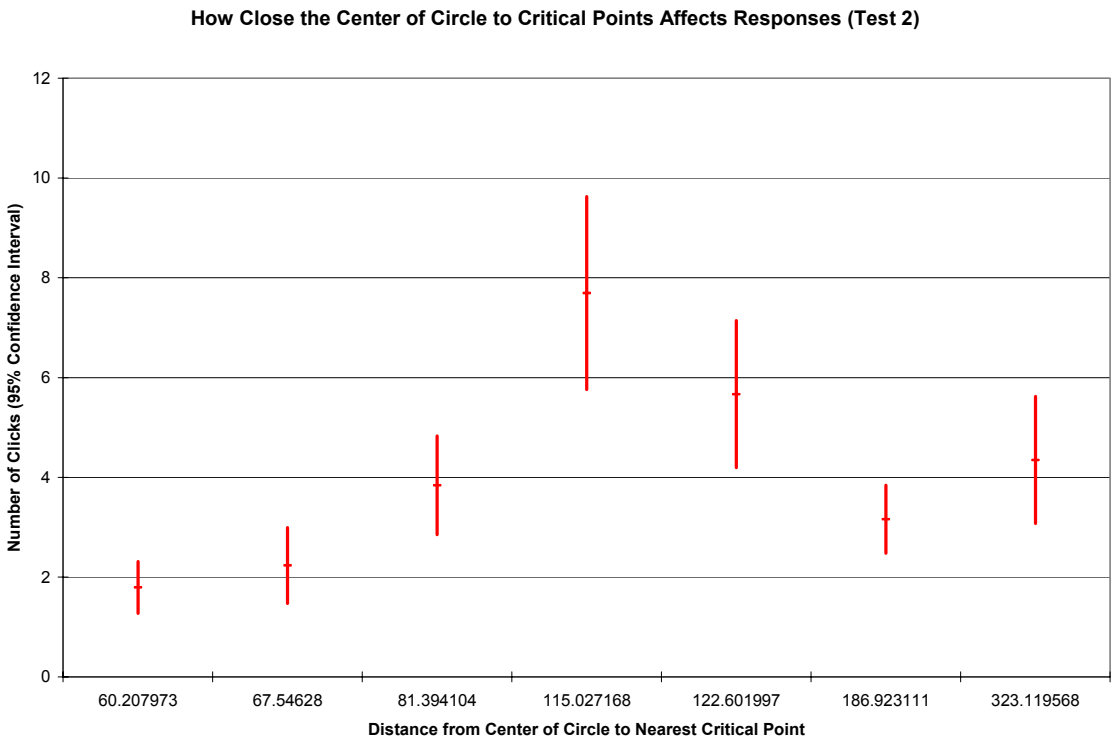

Figure 16: 95% confidence intervals of how close the center of the circle was to the nearest critical point and how this distance affected results for test two.

CHAPTER 7

CONCLUSIONS

No statistical conclusions can be drawn from the graphs. This indicates one of two things. First, there was a problem with the test methodology. Second, visualization methods for unsteady flow fields do not do a good enough job to show the underlying phenomena of the data sets.

The test methodology was flawed in a few ways. First, users did not use the program in front of the programmer. Therefore, the programmer must rely on the users to read all documentation materials provided. From the feedback that was received from users after they used the program (or attempted to use the program in some people's case), people did not fully read the instructions or the documentation. Also, the responses seemed to indicate people became "bored" and did not make as good of a response as they could have in some of the later test cases. One of the best results came from someone who made a comment that he thought the test cases were "like a video game." This user took his time to make the best response possible and the data indicated that his responses were more accurate than most other users. To combat this problem, there needs to be someway to make sure the user understands the instructions and takes his time when using the test framework. Creating a simple quiz the user must complete before he uses the program could solve the problem of not reading the instructions. The quiz would test the user on basic terminology and other aspects of what he is doing to insure he understands what is going to occur. By setting a minimum amount of time the user must spend on a test could help with the

problem of people making too hasty of responses.

Another problem with the test methodology was that the test cases were made by a human. Whether the programmer intends to do so or not, some tests are going to be more difficult than others. This does not matter when comparing overall results for the different visualization methods since the visualization methods are randomly assigned to test cases. However, when comparing how variables affect the results (as shown in the graphs in the Statistical Results section above), this is a problem. For instance, say in reality that the larger the circle radius the easier it is for the user to advect a point to the outline of the circle. If the test cases that involve the larger circle radius are unintentionally made too difficult by the programmer, the result will be that larger circle radii will be harder for the user to advect points to. This accounts for some of the graphs not having a "trend" to them. Take a look at Figure 14 in the section preceding this one (Statistical Results). A circle radius of 150 or 300 shows bad results. However, circle radii of 200 or 350 show very good results. If a trend is expected, there should not be sudden jumps in confidence intervals. A fix for this problem is to have the computer randomly pick the values of the variables.

One last problem with the method of testing is that in certain cases it is difficult for the user to distinguish which way a particle will be advected. Hedgehog is the only visualization method that shows where particles are seeded and which way they go initially from the seed. Color or glyphs can be used for streamlines, pathlines, or streaklines to help indicate the direction of the advection paths. However, LIC is naturally grey scale and is not normally drawn with color. It is also texture based and adding glyphs on top of the textured quad is also unnatural.

It is also possible that even after correcting the test methodology that the responses obtained from users will not give any statistical evidence to draw a conclusion about whether one visualization method is the "best." Since the user is only shown one time slice at a time, he must make a mental note about what is occurring when he changes to another time slice. It has been shown in psychological research that a person can only hold about seven small items in short-term memory at any one time (a small item might be something like a number or a letter) [4]. To get the full understanding of an unsteady flow data set, a person may have to view and comprehend every time slice. Some data sets can contain over 100 time slices. Also, a time slice is a much larger piece of information to store in short-term memory than just a number. Since unsteady flow data is so complicated, it might be the case that no vector field visualization method is as good they are believed to be. It just might be the case that when a person is presented with an unsteady flow, his best way to make a response is just to guess.

CHAPTER 8

FUTURE WORK

As explained in the previous section, the test methodology could be improved upon. If future tests provide statistical evidence that some unsteady flow field visualization methods are better than others, the domain of visualization methods can be expanded. Other methods of interest might be domain deformation, timelines, or IBFV. Also, the 3D vector domain could be explored.

This paper only dealt with how well a user could advect a particle. As mentioned earlier in the paper, other areas of interest include how well a person can identify where critical points are located and how well a person can actually identify what type of critical point exists at a location.

BIBLIOGRAPHY

[1] Brill, M.; Hagen, H.; Rodrian, H.-C.; Djatschin, W.; Klimenko, S.V. 1994. Streamball techniques for flow visualization. In *Proceedings of IEEE Visualization 94*, pages 225-231.

[2] Cabral, B. and Leedom, C. 1993. Imaging vector fields using line integral convolution. In *Proceedings of ACM SIGGRAPH 93*, pages 263-270.

[3] Helman, J.; Hesselink, L. 1990. Surface representations of two- and three-dimensional fluid flow topology. In *Proceedings of IEEE Visualization 90*, pages 6-13, 460.

[4] Hockenbury, D. and Hockenbury, S. *Psychology*. Third Edition. Worth Publishers. 2003, pages 243-244.

[5] Kosara, R.; Healey, C.G.; Interrante, V.; Laidlaw, D.H.; Ware, C. 2003. Visualization viewpoints. In *IEEE Computer Graphics and Applications*, pages 20-25.

[6] Laidlaw, D.H.; Kirby, R.M.; Davidson, J.S.; Miller, T.S.; da Silva, M.; Warren, W.H.; Tarr, M. 2001. Quantitative comparative evaluation of 2D vector field visualization methods. In *Proceedings of IEEE Visualization 01*, pages 143-150.

[7] Laidlaw, D.H.; Kirby, R.M.; Jackson, C.D.; Davidson, J.S.; Miller, T.S.; da Silva, M.; Warren, W.H.; Tarr, M.J. 2005. Comparing 2D vector field visualization methods: a user study. *In IEEE Transactions on Visualization and Computer Graphics*, pages 59-70.

[8] Schroeder, W.J.; Volpe, C.R.; Lorensen, W.E. 1991. The stream polygon – a technique for 3D vector field visualization. In *Proceedings of IEEE Visualization 91*, pages 126-132, 417.

[9] Shen, H.-W. and Kao, D.L. 1997. UFLIC: a line integral convolution algorithm for visualizating unsteady flows. In *Proceedings of IEEE Visualization 97*, pages 317-322, 556.

[10] Wijk, J. Image based flow visualization. 2002. Image based flow visualization. In *Proceedings of ACM International Conference on Computer Graphics and Interactive Techniques 02*, pages 745-754.